BGP

Border Gateway Protocol

IGP vs EGP

- AS - Autonomous System - collection of connected IP routing prefixes under the same administrative entity that presents a common routing policy
- IGP - Interior Gateway Protocol used for exchanging routing table information between gateways within an autonomous system. That group include RIP and OSPF
- EGP - Exterior Gateway Protocol used for exchanging routing table information between gateways from different autonomous systems. That’s currently only BGP

What does it take to become an ISP?

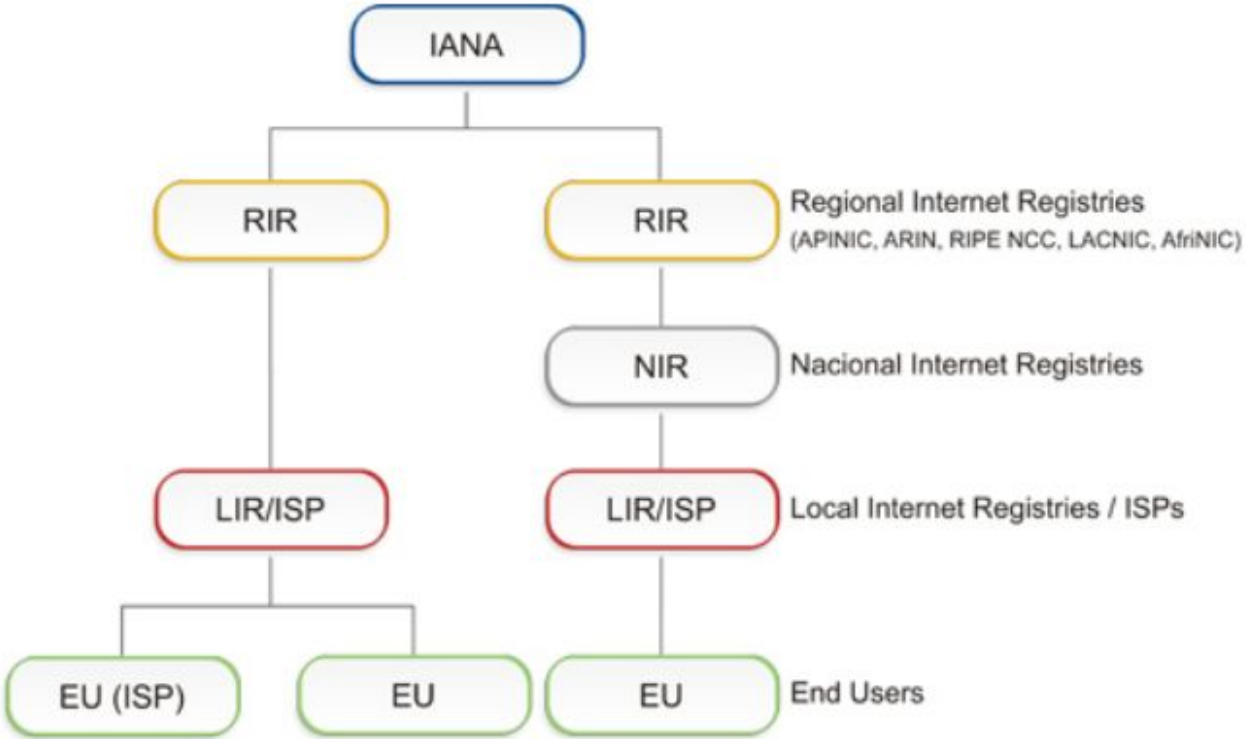
- Presence in appropriate RIR,
- Public ASN assigned,
- Public IP space assigned,
- Someone to peer with upstream

… alright, so where does this journey start?

Institutions governing the IP addresses assignments

- **IANA** (Internet Assigned Number Authority) - takes care of IP address allocation, autonomous system numbers (**ASNs**) allocation, root zone management in DNS and more.
- **RIR** (Regional Internet Registry) - assigns IP blocks to **NIRs** (National Internet Registries) or **LIRs** (Local Internet Registries)

Institutions governing the IP addresses assignments



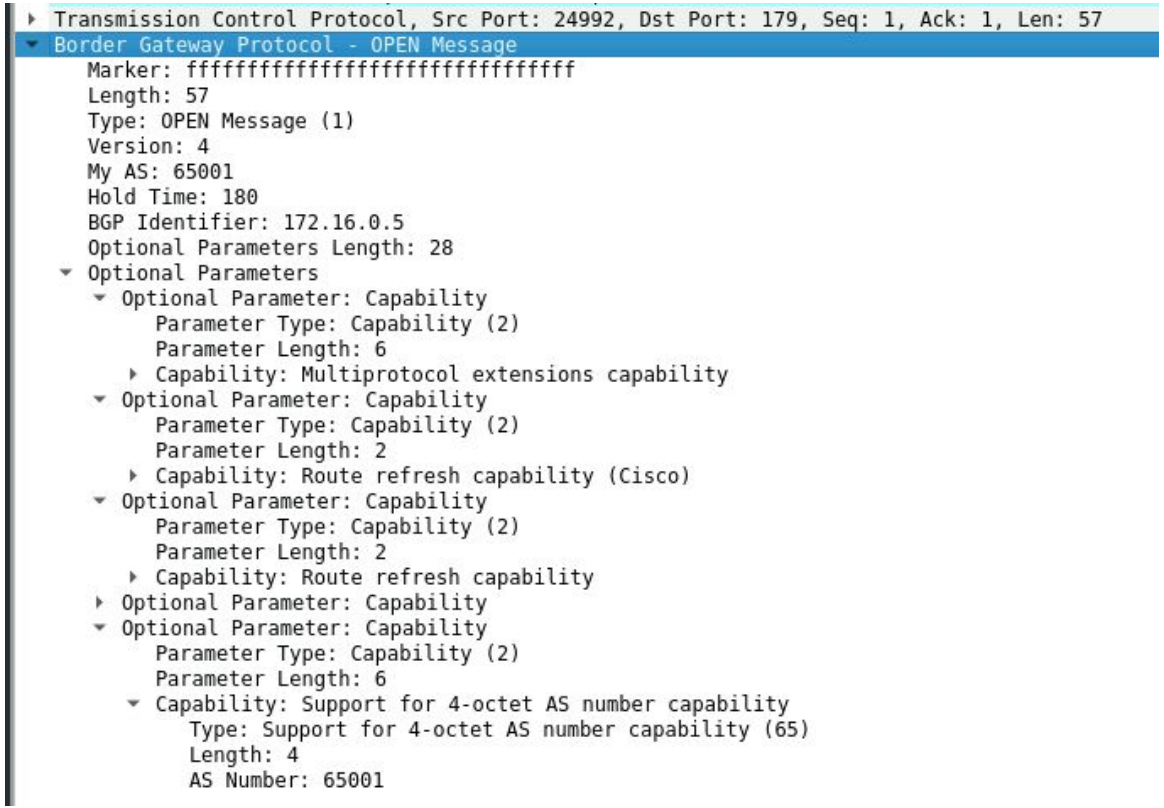
I have got public IP address pool, now what?...

- **BGP** (Border Gateway Protocol) is the only protocol used to take care of the IP reachability across the Internet
- Each entity (end user: company / ISP..) is assigned an **ASN** (Autonomous System Number) which is the number identifying the network under the same management domain:
  - 0 - reserved
  - 1-64495 - public numbers
  - 64494 - 64511 - reserved for documentation
  - 64512 – 65534 - private ASNs
  - 65535 - reserved
  - 4 bytes ASN - extension to public numbers, goes up to 4,294,967,296

BGP peerings

- Routers establish peerings with each other to exchange IP prefixes
- Peering is established over TCP session (BGP uses port 179),
- In the beginning of that process, routers exchange OPEN packets where they list some basic information about themselves and the session itself:
  - Local ASN,
  - Timers (Hold),
  - BGP Identifier,
  - Supported address families (IPv4, IPv6, VPNV4, L2VPN, EVPN…),
  - Supported session capabilities (Route Refresh, 4-byte ASN, MP-BGP extensions...)

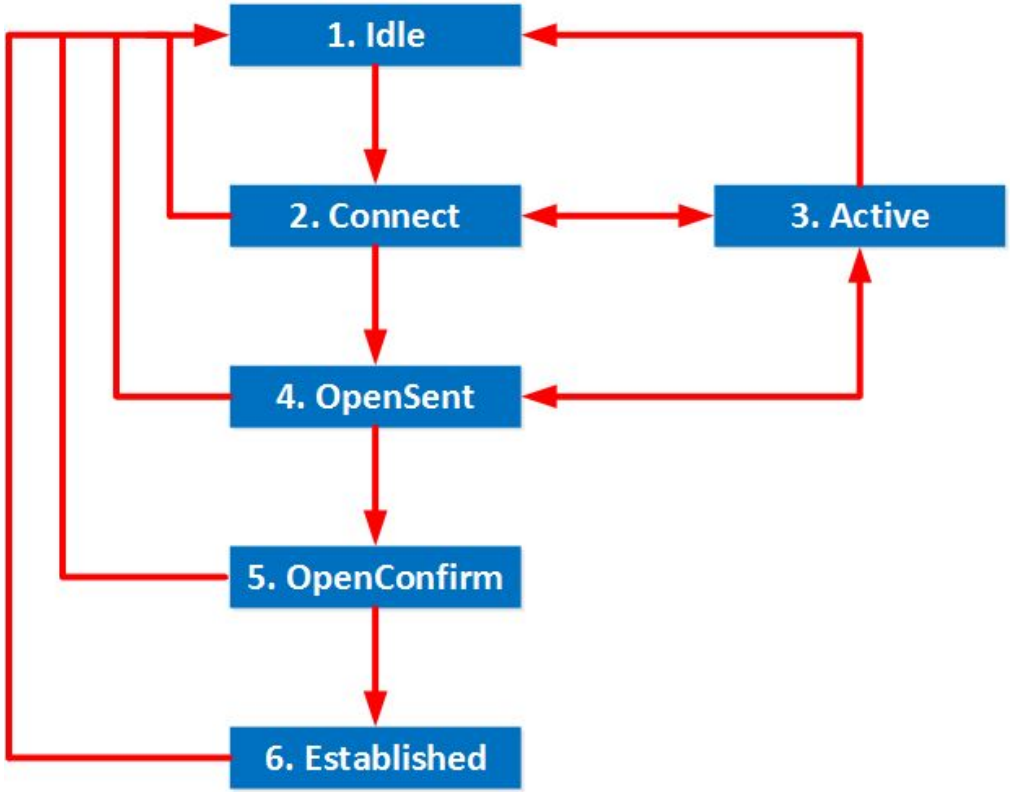
BGP peerings - OPEN message



BGP states

- **Idle** - all incoming connection requests are rejected; waiting for process to start / configured peer is unreachable
- **Connect** - router awaits TCP session to be initiated from peer. If successful, it transitions to the *OpenSent* state. If not, it will try to initiate the session itself and will transition to the *Active* state
- **Active** - router actively tries to get the TCP session established
- **OpenSent** - router sent out *OPEN* message and awaits the same from the remote power. If session parameters are incorrect it will move back to *Idle* state. Otherwise it will send out a *KEEPALIVE* message.
- **OpenConfirm** - router waits for *KEEPALIVE* message. If it doesn't get before the Hold Timer expires or if it gets *NOTIFICATION* message it will get back to *Idle* state.
- **Established** - BGP exchanges *UPDATE*, *KEEPALIVE* and *NOTIFICATION* messages

BGP states


1. Idle
2. Connect
3. Active
4. OpenSent
5. OpenConfirm
6. Established

BGP peerings - session establishment requirements

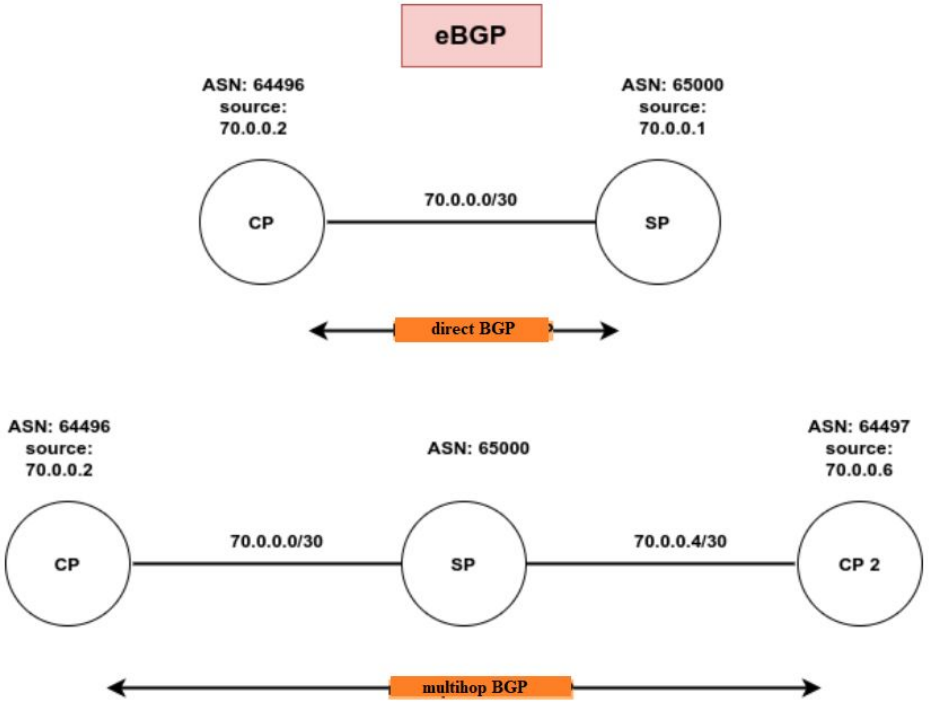
- ASN numbers must match,
- Hold Timer doesn’t have to match - the lower of two is chosen,
- By default, peers have to be directly connected (L3) / one hop away
  - This can be overridden by using *multihop* functionality
- Peers have to be reachable
- BGP Identifiers must be unique
- Authentication parameters must match
  - MD5 password,
  - TCP-AO (Authentication Option) - keys

Types of BGP peerings

- **eBGP** (External BGP) - session between routers in two different ASNs
  - direct (within same L3 network),
  - indirect (multiple hops away)
- **iBGP** (Internal BGP) - session between routers in the same ASN

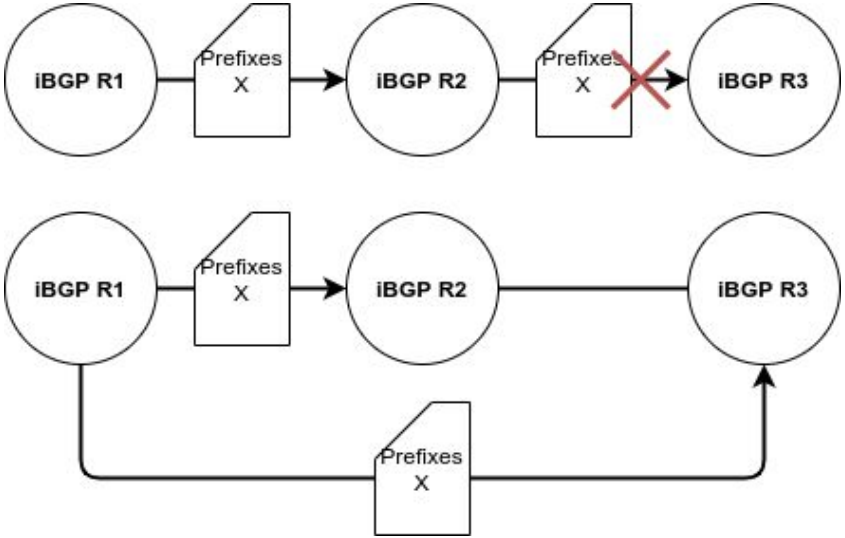
Types of BGP peerings - eBGP

- **eBGP** (External BGP) - session between routers in two different ASNs
  - direct (within same L3 network)
  - indirect (multiple hops away)



Types of BGP peerings - iBGP

- **iBGP** (Internal BGP) - session between routers in the same ASN. Prefixes learn from iBGP neighbors are not advertised to other iBGP neighbors in order to avoid loops in the network. Because of this, full mesh connectivity must be satisfied to establish full reachability

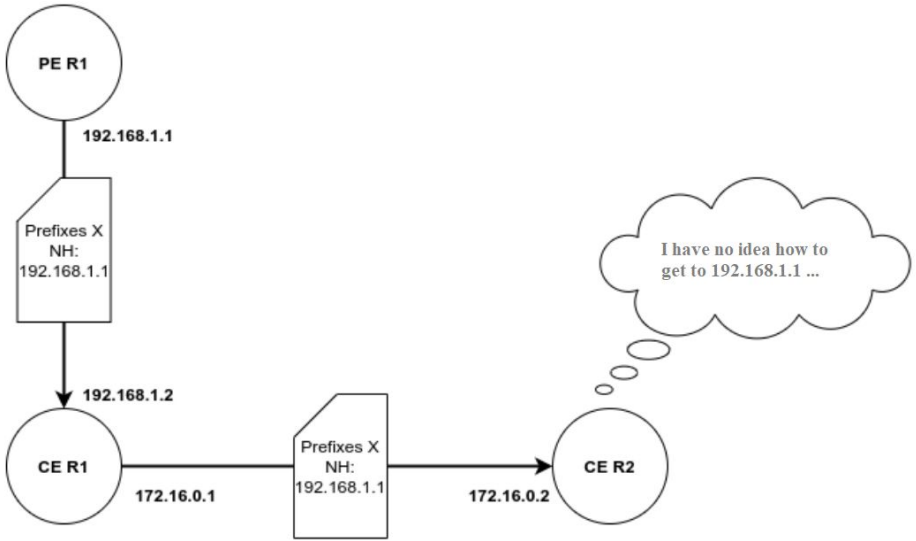


Types of BGP peerings - iBGP

- **iBGP** Router advertises **eBGP**-learned prefixes to its **iBGP** neighbors,
- **iBGP** router advertises **iBGP**-learned prefixes to its **eBGP** neighbors,
- **iBGP** router does not advertise **iBGP**-learned prefixes to its **iBGP** neighbors unless it acts as *Route Reflector*. *This subject is beyond scope of this presentation.*

Types of BGP peerings - iBGP

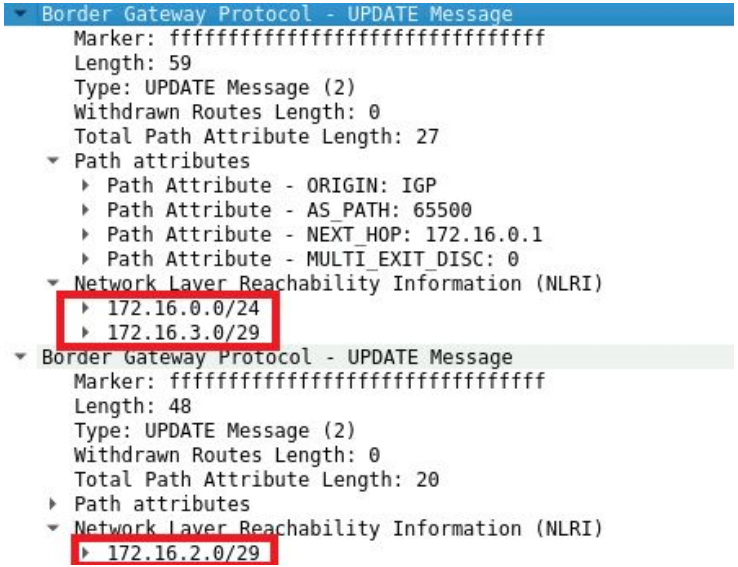
- **iBGP** Routers do not include their ASN in the AS path for the advertised prefixes
- **iBGP** Routers do not change next-hop IP address for the given prefix



This behavior can be overridden by using ***next-hop self*** functionality. Next-hop will be then set to the IP address of the interface which the UPDATE originated from

BGP peerings - UPDATE message

- This type of packet is used to perform the actual exchange of the prefixes within the **NLRI** section (Network Layer Reachability Information):



BGP prefix attributes

Attribute	Category	Description
Next-hop	Well-known, mandatory, non-transitive	Address which the advertised prefix is reachable through
AS Path	Well-known, mandatory, transitive	List of ASNs in the path. The shorter the better.
Origin	Well-known, mandatory, transitive	The source type of the prefix. I (IGP) > E (EGP) > ? (Unknown)
Local preference	Well-known, discretionary	Preference value for prefixes learned from particular neighbor. Used in iBGP - the higher the better.
Originator ID	Optional, non-transitive	Only in iBGP - RID, of the originating router

BGP prefix attributes

Attribute	Category	Description
MED	Optional, non-transitive	Metric internal to the remote peer. The lower the better. By default remote ASN needs to be the same.
Aggregator	Optional, transitive	ANS and IP address of the routera, which originated the aggregated prefix
Community	Optional, transitive	Route tag

BGP prefix selection process

- **Highest local preference**
- **Shortest AS-path**
- **Lowest origin value (I > E > ?)**
- **Lowest MED (by default only for the same peering ASN)**
- **Prefer eBGP-learned prefixes over iBGP-learned ones**
- **For routes learned over iBGP, prefer neighbor with the lowest IGP cost to reach it**
- Prefer prefix from the neighbor with the lowest RID. If they are the same, prefer the currently active prefix.
- For iBGP prefer the shortest cluster length,
- Prefer prefix from the neighbor with the lowest next-hop IP address

BGP prefix selection process - exercise 1

Which one is better for the same prefix?

AS Path	Origin	Local preference	MED	Remote ASN
100 123 245 567	I	100	200	100
100 145 55 67 89	I	200	100	100

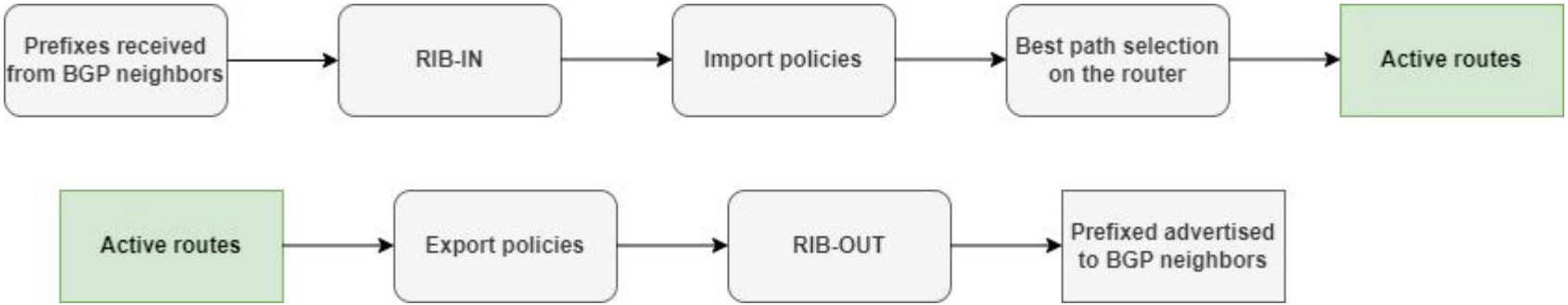
BGP prefix selection process - exercise 2

Which one is better for the same prefix?

AS Path	Origin	Local preference	MED	Remote ASN
123 245 567	I	100	200	123
123 567 245	I	100	100	123

BGP prefixes processing

- BGP keeps information about the prefixes in three main tables:
  - RIB-IN - all known paths,
  - RIB-LOCAL - only active paths,
  - RIB-OUT - paths advertised to peers (*only active ones by default*)



BGP configuration bits (Zebra)

- BGP allows for quite flexible routes manipulation. The basic building blocks to achieve this are the following:
  - **Prefix-lists** - list of prefixes subjected to manipulation,
  - **Route-maps** - powerful tool which allows to define policies that control the redistribution of routes, set route attributes, filter routes, and influence routing decisions.

BGP configuration bits (Zebra) - prefix-list

- List of prefixes with specified number of bits which need to match along with optional specification of netmask length range

```
ip prefix-list PL1 seq 5 permit 10.0.0.0/24
ip prefix-list PL1 seq 10 permit 10.0.1.0/24 le 30
ip prefix-list PL1 seq 15 permit 10.0.2.0/24 ge 25 le 30

ip prefix-list GOOGLE_DNS seq 5 permit 8.8.8.8/32
ip prefix-list GOOGLE_DNS seq 10 permit 8.8.4.4/32
```

BGP configuration bits (Zebra) - prefix-list

- How would the single prefix-list entry to allow *all* the prefixes look like?

BGP configuration bits (Zebra) - prefix-list

- How would the single prefix-list entry to allow *all* the prefixes look like?

```
ip prefix-list ALLOW_ALL seq 5 permit 0.0.0.0/0 le 32
```

BGP configuration bits (Zebra) - route-maps

- Route maps work hand in hand with prefix-lists to find a matching prefix which you would like to change attributes of or influence routing decision for.

```
route-map HIGH_LOCAL_PREF_GOOGLE permit 10
 match ip address prefix-list HIGH_LOCAL_PREF_GOOGLE
 set local-preference 200
```

BGP configuration bits (Zebra) - route-maps

- Route maps work hand in hand with prefix-lists to find a matching prefix which you would like to change attributes of or influence routing decision for.

```
route-map HIGH_LOCAL_PREF_GOOGLE permit 10
 match ip address prefix-list HIGH_LOCAL_PREF_GOOGLE
 set local-preference 200
```

Would that work?

BGP configuration bits (Zebra) - route-maps

Yes!.. and no

- Route-maps have implicit “deny-all” statement at the very end. If given prefix is not matched all the way down through all the statements they it will be denied.

BGP configuration bits (Zebra) - route-maps

Yes!.. and no

- Route-maps have implicit “deny-all” statement at the very end. If given prefix is not matched all the way down through all the statements they it will be denied.

So, how should the presented route-map look like?

BGP configuration bits (Zebra) - route-maps

```
route-map HIGH_LOCAL_PREF_GOOGLE permit 10
 match ip address prefix-list HIGH_LOCAL_PREF_GOOGLE
 set local-preference 200
route-map HIGH_LOCAL_PREF_GOOGLE permit 20
 match ip address prefix-list ALLOW_ALL
```

or even:

```
route-map HIGH_LOCAL_PREF_GOOGLE permit 10
 match ip address prefix-list HIGH_LOCAL_PREF_GOOGLE
 set local-preference 200
route-map HIGH_LOCAL_PREF_GOOGLE permit 20
```

… because “no match” means “match all”